

---

# Arrays

In this laboratory session you will:

1. Learn how to declare, create and initialize arrays
2. Learn how to manipulate arrays with loops
3. Investigate how arrays are passed to methods
4. Investigate arrays of primitive values and arrays of objects
5. Learn how `String[] args` is used in Java applications
6. Investigate an array of arrays

## One-Dimensional Arrays

In Java, an array is an object that can contain multiple values of the same data type. Each of these values is an element of the array. An array that contains `int` values has data type `int[]`. An array that contains `char` values has data type `char[]`. An array of `char` named `grades` is declared with the statement

```
char[] grades;
```

The statement above merely declares `grades` to be a variable of type `char[]`. It does not create an array nor does it assign a value to `grades`. To create an array and assign that array to the variable `grades`, one uses a statement such as

```
grades = new char[5];
```

which is an assignment statement in which the `new` operator is used to create the array that is then assigned to the variable `grades`. Note that the size of the array is specified within brackets. In particular, the above statement creates an array capable of holding 5 integers. The capacity of an array is called the *length* of the array. That is, the array above is said to have length 5.

The declaration, creation, and assignment of an array can be combined into a single statement such as

```
int[] temperature = new int[24];
```

which declares an array of `int` named `temperature` and assigns to it a new array of `int` that can hold twenty-four `int` values.

Later in the program, individual elements of an array are referenced by the array name followed by the index of the particular element within brackets. As with strings, index values start at 0; that is, the first element has index 0, the second has index 1, and so on. Thus, the array `grades` as defined above is an array of length 5 whose elements are referenced by `grades[0]`, `grades[1]`, `grades[2]`, `grades[3]`, and `grades[4]`. In particular, the statements

```
grades[0] = 'B';
grades[1] = 'C';
```

assign the characters B and C to the first two elements of the array `grades`. Moreover,

```
temperature[23] = temperature[9] + temperature[12];
```

assigns the sum of the tenth and thirteenth elements (at indices 9 and 12 respectively) of `temperature` to the 24th element (at index 23).

The elements of an array can be initialized at the time the array is declared by following the declaration with an `=` symbol and a list of the values to be assigned within braces. Thus,

```
char[] grades = {'B', 'C', 'A', 'C', 'D'}
```

not only establishes an array named `grades` but also assigns the values B, C, A, C, and D to the array's elements. When declaring, creating, and initializing an array in this manner, the length of the array is determined by the number of values listed between the braces.

While an array is an object, there are no existing methods that may be invoked on an array. However, every array has one `public` instance variable named `length` of type `int` that is assigned the length of the array when the array is created. Thus, `grades.length` is 5 and `temperature.length` is 24.















### Experiment 10.3

This experiment investigates what happens when an array and an element in an array are passed to a method. The driver class `J10E03` creates two arrays and an object of type `E03`. The `E03` class describes only a single method that is passed two `int` values and one array of `int`. Before and after invoking the `fun` method on the `E03` object, data values are printed.

**Step 1.** Execute the program `J10E03.java`. Record the results.

```
class J10E03
{
    public static void main(String[] args)
    {
        int[] arrayA = {2,3,4};
        int[] arrayB = {20, 30, 40};
        int j = 5;
        E03 tester = new E03();

        System.out.println(
            "Before tester.fun(numbers):" +
            "\n    j = " + j +
            "\n    arrayA[0] = " + arrayA[0] +
            "\n    arrayB[0] = " + arrayB[0]);

        tester.fun(j, arrayA[0], arrayB);

        System.out.println(
            "After tester.fun(numbers):" +
            "\n    j = " + j +
            "\n    arrayA[0] = " + arrayA[0] +
            "\n    arrayB[0] = " + arrayB[0]);
    }
}

class E03
{
    public void fun(int x, int y, int[] a)
    {
        x = 100;
        y = 200;
        a[0] = 300;
    }
}
```

---



---



---



---



---



---



---



## Arrays of Objects

The elements in the arrays we have studied so far have been of primitive types. The elements of an array may also be references to objects. For example, each element of the array `animals` as defined by

```
String[] animals = {"cat", "dog", "mouse"};
```

contains a reference to a `String`. That is, `animals[0]`, `animals[1]` and `animals[2]` are references to the objects "cat", "dog", and "mouse", respectively.

### Experiment 10.4

**Step 1.** Compile and execute the program `J10E04.java`. Record the results.

```
class J10E04
{
    public static void main(String[] args)
    {
        String[] animals = {"cat", "dog", "mouse"};
        for(int j = 0; j < animals.length; j++)
        {
            System.out.print(animals[j] + " ");
        }
        System.out.println();
    }
}
```

---



---



---



---



---



---

**Step 2.** Modify the program in Step 1 by adding the following loop before the existing loop.

```
for(int j = 0; j < animals.length; j++)
{
    System.out.println(animals[j].toUpperCase());
}
```

Predict the results.

---



---



---



---



---





## Explaining String[] args

The header of the main method

```
public static void main(String[] args)
```

indicates that the main method expects to receive an array of type `String`. The elements of the array are passed at the time the command is given to execute the program. Your teacher will give you instructions for running a Java program with *command line arguments* on your system. Inside the program, the arguments are accessed using `args[0]`, `args[1]`, etc.

### Experiment 10.5

**Step 1.** Compile and execute the program `J10E05.java` using no command line arguments. Record the results.

```
class J10E05
{
    public static void main(String[] args)
    {
        System.out.println(
            "Investigating args: \n" +
            "    args.length is " + args.length);
    }
}
```

---

---

---

---

---

---

---

---

---

---

---

---

**Step 2.** Execute the program again using your first name as a command line argument. Record the results.

---

---

---

---

---

---

---

---

---

---

---

---

**Step 3.** Execute the program again using your first and last names as command line arguments. Record the results.

---

---

---

---

---

---

---

---

---

---

**Step 4.** Modify the program in Step 1 to print `args[0]`. Execute the program using your first name as a command line argument. Record your modifications. Record the results.

---

---

---

---

---

---

---

---

---

---

**Step 5.** Modify the program in Step 4 to print `args[1]` also. Execute the program again using your first and last names as command line arguments. Record the results.

---

---

---

---

---

---

---

---

---

---

**Step 6.** Execute the program in Step 5 using only your first name as a command line argument. Record the results.

---



---



---



---



---



---



---



---



---



---



---



---



---



---



---



---

### Arrays of Arrays (Multi-dimensional arrays)

The arrays that we have studied so far are called one-dimensional arrays. In Java, multi-dimensional arrays are created as arrays of arrays. For example,

```
int[][] grid = new int[][3];
```

declares and creates an array of length 3 whose elements are arrays of `int`. The array elements do not yet exist. The elements in `grid` may be arrays of different lengths. For example, the code

```
grid[0] = new int[5];
grid[1] = new int[2];
grid[2] = new int[4];
```

creates three new `int` arrays with lengths five, two and four and assigns these arrays, respectively to `grid[0]`, `grid[1]`, and `grid[2]`. The statement

```
grid[0][4] = 9;
```

assigns 9 to the fifth element of the array stored in `grid[0]`. If the elements of the arrays are known initially, the entire contents of `grid` and each of the arrays that it contains may be declared, created, and initialized in a single statement such as

```
int[][] grid = {{1,3,5,7,9}, {0,1}, {2,4,6,8}};
```

that declares and creates an array `grid` that contains three arrays of `int`. Each of these arrays is created and initialized. The entry `grid[0]` is an array of length five whose elements are 1, 3, 5, 7, and 9. The entry `grid[1]` is an array of length two with elements 0 and 1. And, the entry `grid[2]` is an array of length four with elements 2, 4, 6, and 8.





## Post-Laboratory Problems

- 10.1.** Write a program that declares and creates an array of 50 `int` values. Then write a loop to do each of the following:
- initialize the elements of the array to the numbers 1, 3, 5, 7, . . . , 97, 99
  - print the values in the array in a chart with five numbers per row
  - find the sum of the numbers in the array
  - change the numbers in the array to the numbers 2, 4, 6, 8, . . . , 98, 100
  - print the values in the array backwards in a chart with five numbers per row
  - find the sum of the squares of the numbers in the array
- 10.2.** Write a program that applies the insertion sort algorithm (see Chapter 4 of *Computer Scienc: An Overview*) to sort the array containing the `int` values 8, -2, 5, 0, -1, 9, 5, 3, -1, 5, 2.
- 10.3.** Write a program that applies the insertion sort algorithm (see Chapter 4 of *Computer Scienc: An Overview*) to sort an array containing ten user-entered `String` elements. To compare strings use the `String` method
- ```
public int compareTo(String s)
```
- which lexicographically compares this `String` to `s` and returns
- o a negative number if this `String` comes before `s`
  - o zero if this `String` is identical to `s`
  - o a positive number if this `String` comes after `s`
- 10.4.** Write a program that uses an array of `String` storing "zero", "one", . . . , "nine" to print the word corresponding to a user-entered integer between 0 and 9.
- 10.5.** Extend the program written in 10.4 using a second array of `String` containing "ten", "eleven", . . . , "nineteen" to handle user-entered integers between 0 and 19.
- 10.6.** Extend the program in 10.5 using a third array of `String` containing "twenty", "thirty", . . . , "ninety" to handle user-entered numbers between 0 and 99.
- 10.7.** Extend the program in 10.6 (no more arrays are needed) to handle integers from 0 to 999.
- 10.8.** Suppose a class has five students: Bob, Ann, Joe, Pat, and Ray. Also, each of the students has three test scores. Bob: 85, 77, 92. Ann: 65, 77, 70. Joe: 78, 85, 82. Pat: 70, 95, 82. Ray: 73, 79, 81.
- Declare, create and initialize two arrays:
- o An array of `String` storing the student names
  - o An array of arrays of `int` storing the five sets of three test scores
- Print the following information:
- o For each test, print the scores
  - o For each student, print the student's test scores
- Print the following averages
- o The average of the scores on each of the three tests
  - o The test average for each of the five students
- 10.9.** Write a program that computes the matrix product of two matrices.